République Algérienne Démocratique et Populaire

Ministère de l’Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie Houari Boumediene

Faculté d’Informatique

Département d’intelligence artificielle

**Rapport du Mini projet n°2**

***Module***

Algorithmes Avancé et Complexité

***Spécialité***

Informatique Visuelle

**Rapport réalisé par :**

* **Etudiant :** SAYOUD Lynda

**Matricule :** 191931031844

**Groupe :** 01

* **Etudiant :** DRAI Said

**Matricule :** 191931029353

**Groupe :** 01

**Partie I**

Développement de l’algorithme et du programme du problème des tours de Hanoi avec le langage C

1. **L’algorithme récursif des tours de Hanoi :**

**Procédure** Hanoi\_tours (N, dep,arr,inter)

**Entrée** N :entier ; //Nombre de disques à déplacer

dep, inter, arr :caractère ; //les tiges par example ‘A’, ‘B’,’C’

**Début**

**Si** (N=1) **alors** 1

Ecrire(‘ Disque 1 déplacé de ‘ dep ‘ vers ‘arr ) ; 2

**Sinon**

*//Déplacer les n-1 disques de A vers B*

Hanoi\_tours(N-1, dep, inter, arr) ; 3

*//Déplacer le n eme disque de A vers C*

Ecrire(‘Disque’ N’ déplacé de ‘dep’vers’ arr) ; 4

*//Déplacer les n-1 disques de B vers C*

Hanoi\_tours(N-1, inter, arr, dep) ; 5

**Fsi ;**

**Fin.**

**Algorithme Hanoi ;**

**Var**

nbDisques :entier ;

**Début**

**Répéter** 6

Lire(nbDisques) ; 7

**Jusqu’à(**nbDisques >=1**) ;**

*// A, B et C sont les tiges(tours ou piquets)   
 //A : piquet initiale, B : Piquet intermédiaire, C : piquet arrivé*

**Hanoi\_tours(**nbDisques,’A’,’C’,’B’**) ;** 8

**Fin.**

1. **1. Complexité temporelle**

Soit **N** le nombre d’instructions de l’algorithme Hanoi (ci-dessus), fi (i=1. . . N) la fréquence d’exécution de l’instruction Ii (i=1. . . N), et F la somme de ces fréquences.

On a : L = 8 instructions.

1. On a pour l’algorithme principal Hanoi :

f1=1 ;  
f2=1 ;

f3=1 + DH(n) ; 1 Operation pour l’instruction d’appel de la fonction Hanoi\_tours (n, A, C, B)

//DH(n) mesure la complexité temporelle de la fonction Hanoi\_tours (n, A, C, B)

Donc, F est :

F(n)=f1 + f2 + f3→ F(n)=1+1+(1+DH(n)) → F(n)= 3+DH(n) ………….(1)

2- Pour le calcul de DH(n), on procède comme suit :

2.1- On calcule les fréquences des instructions de la fonction Hanoi\_tours (. . . ) (instructions 1 à 5) :

f1=1 ;

f2=1 ;

f3=1 +DH(n-1) ;

f4=1 ;

f5=1 +DH(n-1) , 1 opération pour l’instruction d’appel de la fonction Hanoi\_tours (n, A, C, B)

*//DH(n-1) mesure la complexité temporelle de la fonction Hanoi\_tours (n-1, A, B, C)*

2.2- On calcule la somme de ces fréquences :

DH(n)=∑5i=1 fi = f1+f2+f3+f4+f5→ DH(n)=1+1+(1 +DH(n-1) )+1+(1 +H(n-1))

**→ DH(n)=2H(n-1)+5 …….(2)**

Et DH(0)=1 *// équation de récurrence*

si n=0 alors la fonction Hanoi\_tours(. . . ) exécute juste le test (n=1)

2.3- On résout l’équation de récurrence (2) avec la méthode de substitution comme suit : DH(n)=2DH(n-1)+5

→ DH(n)=2 (2DH(n-2)+5)+5

→ DH(n)= 22 DH(n-2)+21\*5+5

→ DH(n)= 22(2DH(n-3)+5)+21\*5+5

→ DH(n)= 23 DH(n-3)+22\*5+21\*5+5

→. . .

→ DH(n)=2nDH(n-n)+2n-1 \*5+ 2n-2\*5+……..+20\*5

→ DH(n)= 2nDH(n-n)+5(2n-1 + 2n-2+……..+20)

→ DH(n)= 2nDH(0)+5(2n-1 + 2n-2+……..+20)

→ DH(n)= 2n\*1+5((2n-1)/(2-1))= 2n+5(2n-1)=6\*2n -5……..(3)  
La relation (3) représente la solution en notation exacte de l’équation de récurrence (2).

Et donc:

DH(n)= 6 ∗ 2n − 5 → DH(n)≤ 6∗ 2n − 5 + 5 = 6 ∗ 2n .

→ DH(n)= **Θ (2n).**

DH(n) en notation asymptotique.  
 On conclut alors que la fonction Hanoi\_tours (. . . ) a une complexité temporelle **exponentielle**.   
 3- On calcule enfin la fonction F(n) en y remplaçant la solution DH(n) dans la relation (1)

F(n)=3+ DH(n) → F(n)=3+(6∗ 2n − 5) = 6∗ 2n – 2… (5)

Donc : F(n)=6 ∗ 2n −2   
→ F(n)≤ 6 ∗ 2 n − 5 + 5 =6 ∗ 2n   
→ F(n)= **Θ (2n).** ….(6)

**Conclusion :**  
l’Algorithme du problème des « tours de Hanoi » a une complexité temporelle **exponentielle**. En conséquence :

**CT(n)** = F(n) = 6 ∗ 2n – 2. *en notation exacte*   
**CT(n)** = F(n) = **Θ (2n).** *en notation asymptotique*

***2.*Complexité Spatiale :**

Après le calcul de la complexité spatiale de l’algorithme d’hanoi :

On a : **CS(𝑛) =** 6\*2n -n +1. *en notation exacte*  
 **CS(𝑛) = Θ (2n).** *en notation asymptotique*

1. **L’algorithme en C :**

#include<stdio.h>

#include<stdlib.h>

void toursHanoi(int N, char Dep, char Arr, char Inter)

{

//N -> nombre de disques à deplacer

if (N == 1)

printf("Disque 1 de %c a %c \n" , Dep , Arr);

else {

// du tour de depart vers l'arrivée

toursHanoi(N - 1, Dep, Inter, Arr);

printf("Disque %d de %c a %c \n", N , Dep ,Arr);

// du tour intermidiaire à l'arrivé

toursHanoi(N - 1, Inter, Arr, Dep);

}

}

main() {

int N;

do {

printf("\n-Entrer le nombre de disques:");

scanf("%d",&N);

}while(N<1);

//Les Tours sont A B C

toursHanoi(N, 'A', 'C', 'B');

}

1. **Les temps d’exécution T :**

On complète le tableau suivant :

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **n** | 5 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 20 |
| **T(s)** | 0.006000 | 0.441000 | 0.8950 | 1.707000 | 3.46521 | 7.0343 | 14.0687 | 28.1375 | 57.11 | 477.7 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **n** | 30 | 35 | 40 | 45 | 50 |
| **T(s)** | …. | …. | ….. | ….. | ….. |

1. **Programme de mesure de temps :**

**void mesurer\_temps(int N)**

**{**

**int T[]={**0.006000, 0.441000, 0.8950, 1.707000, 3.46521, 7.0343}

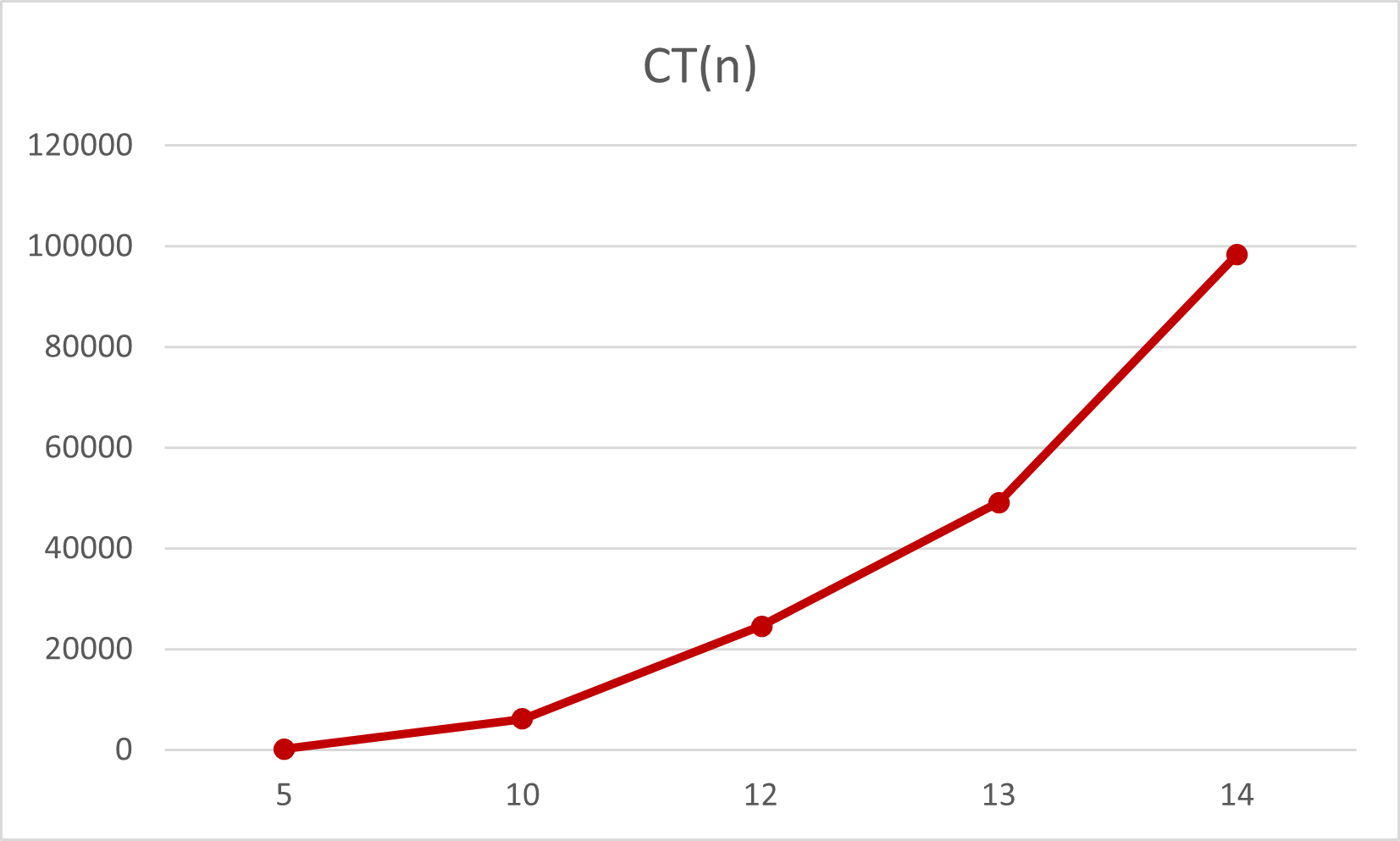
**int CT ;**

**CT=(6\*(2\*\*N))-2 ;**

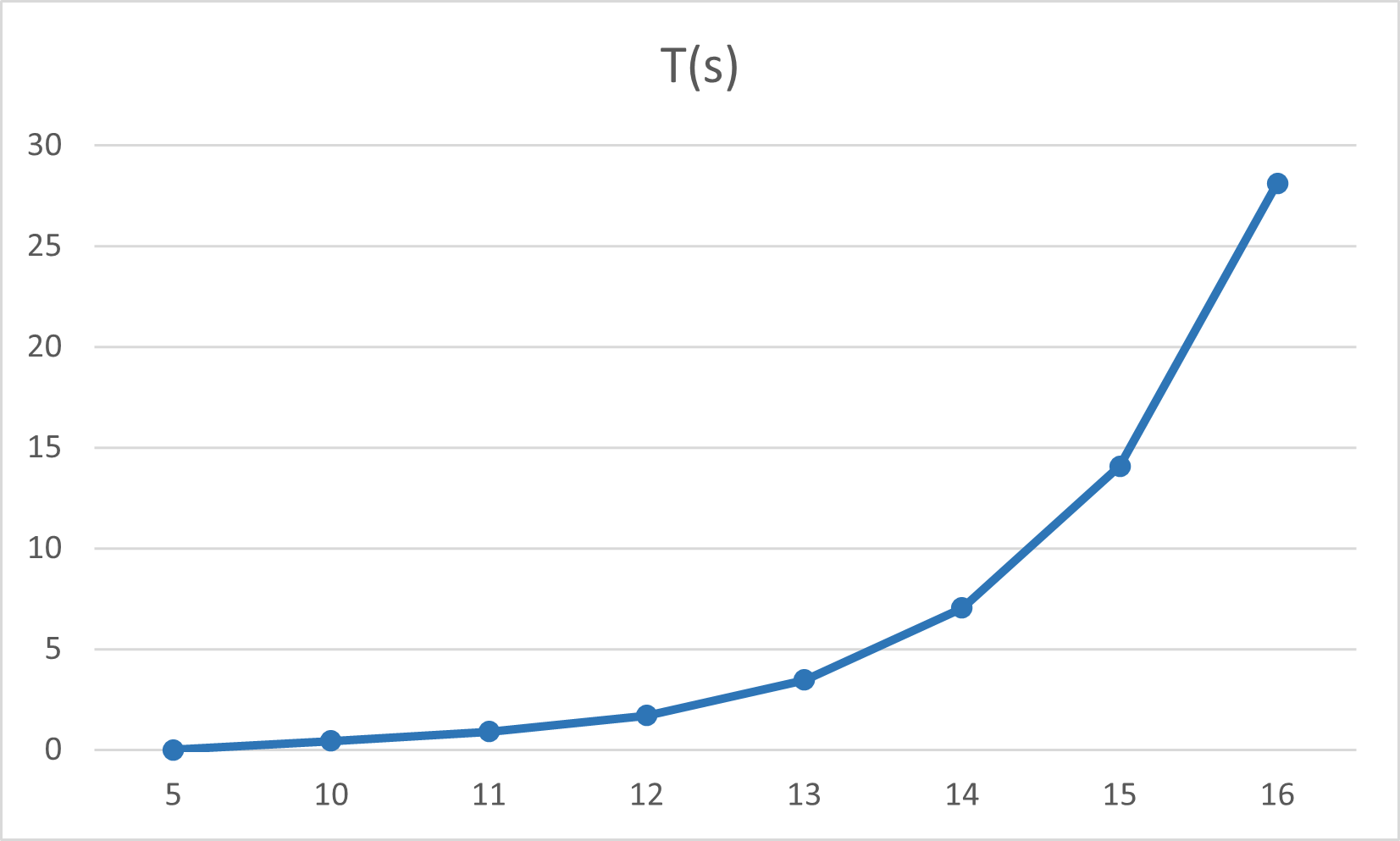
**printf(‘’Complexité théorique pour %d disques : %d ‘’,N,CT) ;**

**}**

1. **Graphes :**
2. Graphe **GCT(n)** des variations de la complexité temporelle CT(n) en fonction de N :



1. Graphe **GT(n)** des variations du temps d’exécution T(n) en fonction de N :



1. **Interprétation des résultats :**

**7.a.** **Comparaison des mesures de temps d’exécution avec le pire et meilleur cas :**   
Dans cet algorithme, nous n’avons pas de complexité au pire et meilleur cas , nous comparons alors la complexité exacte obtenu et ce qui est claire à partir du graphe que les temps d’exécution suivent la même évolution, exponentielle et elle est en O(2n ).

**7.b.** **Remarque et déduction d’une fonction T(n) reliant n au temps d’exécution :**  
 On remarque que les temps d’exécution sont approximativement multipliés par 2 lorsque N est incrémenté de 1. (quand on compare chaque nbr avec son suivant)  
Exemples :   
N1 = 10 ⇒ T1 = 0.441000  
N2 = 11 ≈ N1 + 1 ⇒ T2 = 0.8950 ≈ 2 ∗ T1   
Aussi N1 = 12 ⇒ T1 = 1.707000 N2 = 13 ≈ N1 + 1 ⇒ T2 = 3.46521≈ 2 ∗ T1   
On en déduit que le temps d’exécution est proportionnel à N, ce que l’on peut représenter par la formule suivante :   
 **T(N + 1) = 21 ∗ T(N)** *pour tous N ∈ [0 − 64]*   
(x étant la tangente d’un point sur le graphe). Nous ne pouvant pas généraliser car les tests que nous avons faits n’englobent pas toutes les valeurs possibles.

**7.c.** **Comparaison de la complexité théorique et expérimentale :**

Dans le cas des données de l’échantillon ci-dessus, la complexité expérimentale est du même ordre de grandeur que la complexité théorique, et Donc **Le modèle théorique est conforme aux mesures expérimentales.**

**Partie II**

Développement de l’algorithme et du programme du problème des tours de Hanoi avec le langage Java

**3) L’algorithme en Java :**import java.io.\*;

import java.math.\*;

import java.util.\*;

import java.time.Duration;

public class Hanoi\_Tours

{

static void Hanoi(int n, char Dep,char Arr, char Inter)

{

if (n == 1)

{

System.out.println("Disque " + n + " de "+ Dep + " a "+ Arr);

}

else

{

Hanoi(n - 1, Dep, Inter, Arr);

System.out.println("Disque " + n + " de "+ Dep + " a "+ Arr);

Hanoi(n - 1, Inter, Arr, Dep);

}

}

public static void main(String args[])

{

int N;

do {

Scanner sc = new Scanner(System.in);

System.out.println("\n-Veuillez saisir le nombre de disques:");

N = sc.nextInt();

}while(N<=1);

double start, end,time;

start = System.currentTimeMillis();

Hanoi(N,'A','C','B');

end = System.currentTimeMillis();

time = (end - start) / 1000F;

System.out.println("\n-Temps d'execution: "+ time +" secondes");

}

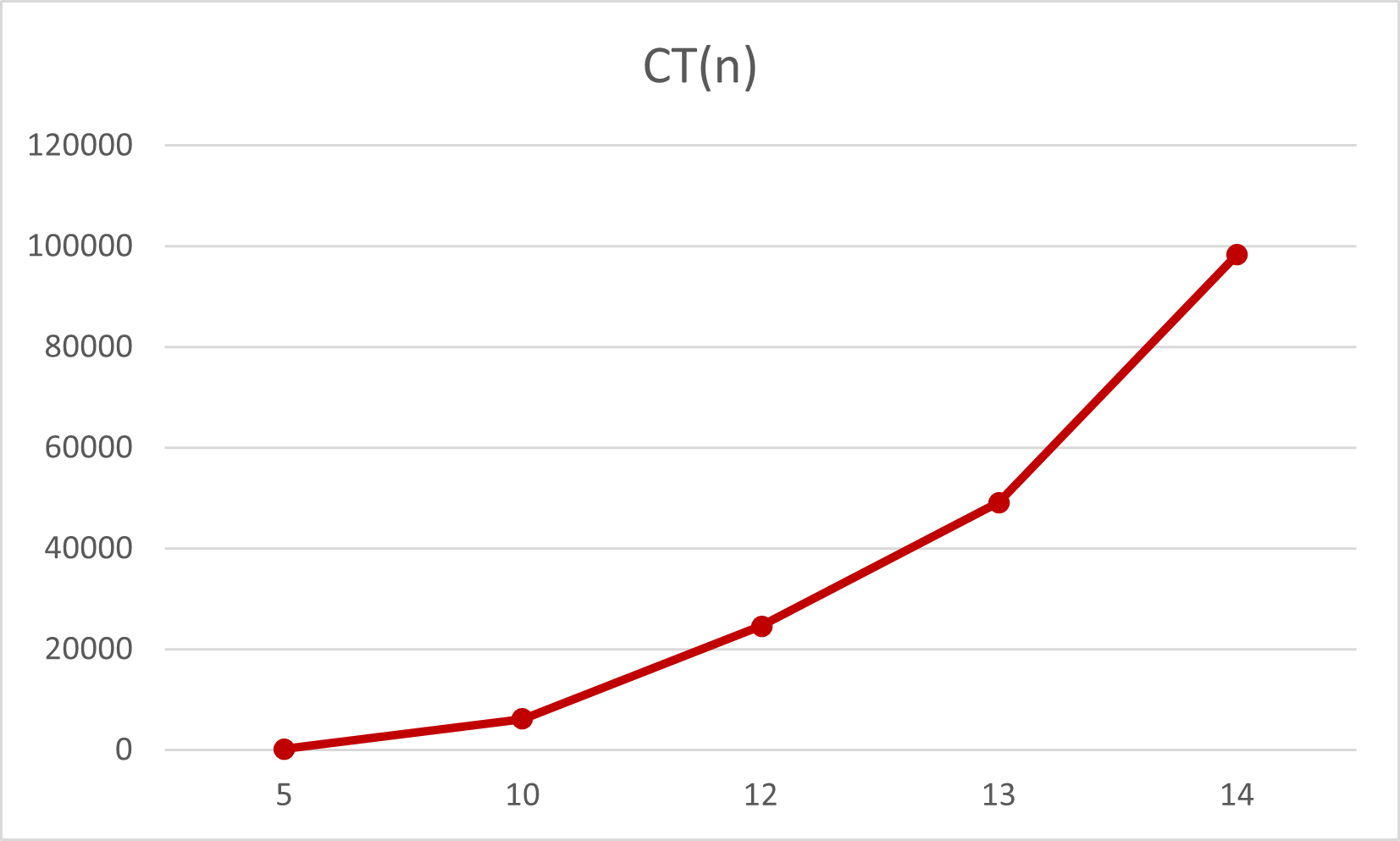
}

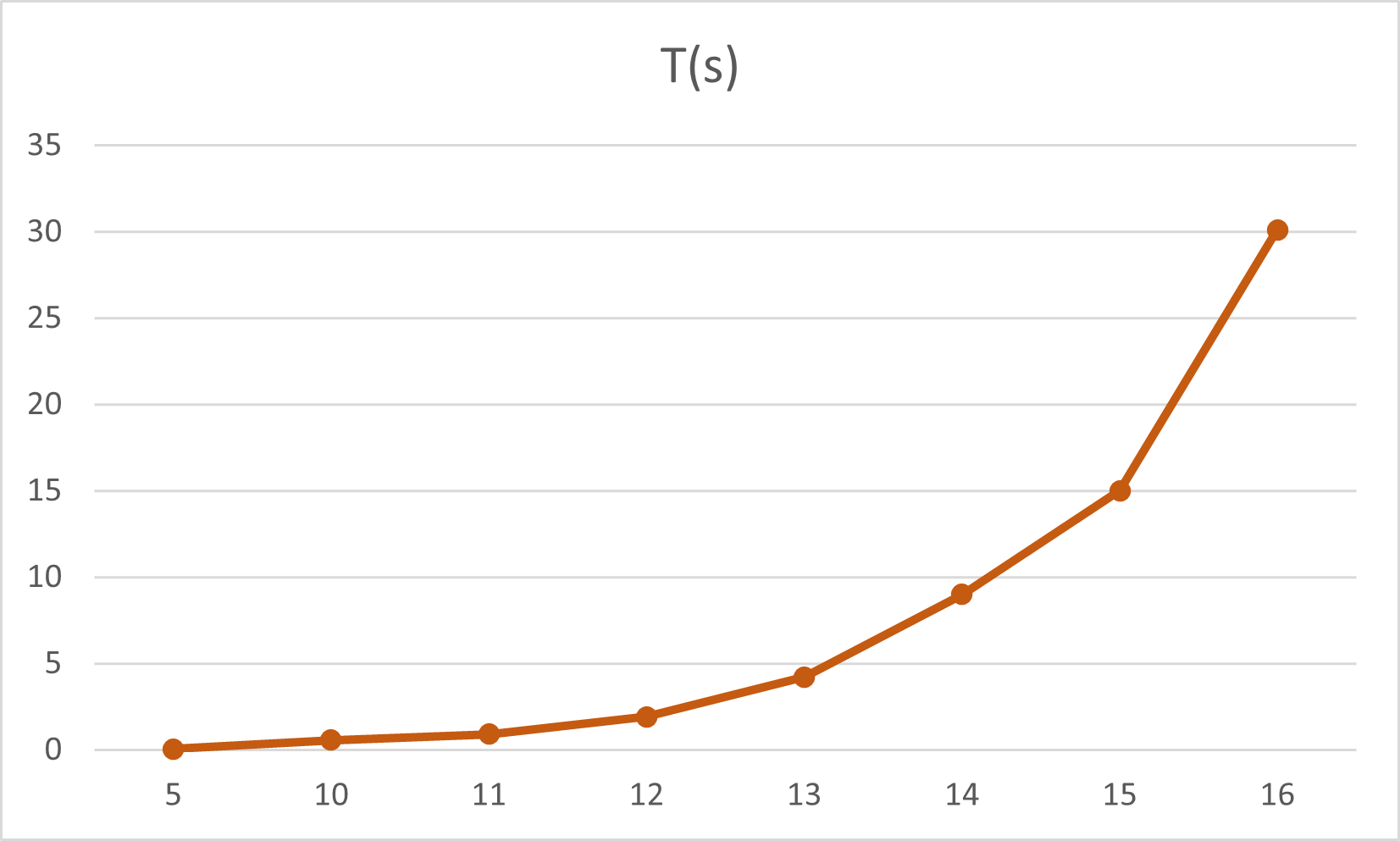
1. **Les temps d’exécution T :**

On complète le tableau suivant :

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **n** | 5 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| **T(s)** | 0.035 | 0.565 | 0.907 | 1.896 | 4.238 | 9.0001 | 15.028 | 30.125 | 75.3125 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **n** | 20 | 30 | 35 | 40 | 45 | 50 |
| **T(s)** | 1176.3409 | …. | …. | ….. | ….. | ….. |

1. **Graphes :**
2. Graphe **GCT(n)** des variations de la complexité temporelle CT(n) en fonction de N :  
     
     
     
   
3. Graphe **GT(n)** des variations du temps d’exécution T(n) en fonction de N :



**Partie**

Conclusion Générale

On note que les temps d’exécution en C sont plus petits que celles du java, autrement dit, l’exécution de l’algorithme de Hanoi en c est rapide qu’en java et cela revient à le fait que le langage C est un langage de bas-niveau et qui proche de langage machine donc une fois compilé il sera exécuté par contre le java est un langage de plus haut niveau il va être d’abord interprété et ensuite traduit en langage machine et ce qui prend un temps.